CLAIMS

We claim:

1. A method for managing a user interface in a multithreaded computing environment, the user interface comprising a plurality of user interface elements wherein a first user interface element in the plurality of user interface elements corresponds to a first application, wherein furthermore the first application having a first thread having control over the first user interface element, method comprising the steps of:
signaling a hung state for the first application; (if there is a delay greater than a predetermined threshold in handling events queued in an event queue for handling by the first application);
creating a ghost user interface element, with a ghost thread, responsively to a hung signal indicating the first application is in the hung state wherein the ghost user interface element replaces the first user interface element in the user interface;
placing a high priority special event in the event queue for handling by the first application wherein handling of the special event generates a activity-detect message; and
detecting the wakeup message and responsively to the activity-detect message replacing the ghost user interface element by the first user interface element.

2. The method of claim 1 wherein the step of signaling includes signaling a hung state if the first application does not handle an event in an event queue for at least a predetermined duration.

3. The method of claim 1 wherein the step of detecting the activity-detect message further comprises releasing, responsively to the activity-detect message, resources used by the ghost user interface element.

4. The method of claim 1 wherein the step of creating the ghost user interface element includes creating the ghost thread, which, in turn, creates the ghost user interface element.

5. The method of claim 4 wherein the step of creating the ghost thread includes determining if a ghost thread exists and creating a ghost thread if the ghost thread does not exist.

6. The method of claim 1 wherein the step of creating the ghost user interface element includes creating the ghost user interface element in the area occupied by the first user interface element.

A13

7. The method of claim 6, the method further having the step of directing input events corresponding to the area covered by the ghost user interface element, which includes at least some of the area covered by the first user interface element, to the ghost thread.

8. The method of claim 7, the method further having the step of having the ghost thread handle at least one event corresponding to the area covered by the ghost user interface.

9. The method of claim 7 having the step of having the ghost thread cache at least one event corresponding to the area covered by the ghost user interface to the ghost thread for handling by the first application.

10. The method of claim 7, the method further having the step of having the ghost thread handle at least one event corresponding to a minimization operation in the area covered by the ghost user interface element.

11. The method of claim 7, the method further having the step of having the ghost thread handle at least one event corresponding to a resizing operation in the area covered by the ghost user interface element.

12. The method of claim 7, the method further having the step of having the ghost thread handle at least one event corresponding to a close operation in the area covered by the ghost user interface element.

13. The method of claim 7, the method further having the step of having the ghost thread handle at least one event corresponding to a move operation in the area covered by the ghost user interface element.

AB

14. The method of claim 7, the method further having the step of having the ghost thread cache at least one event corresponding to a keyboard input corresponding to the ghost user interface element.

15. The method of claim 7, the method further having the step of having the ghost thread handle at least one event corresponding to a keyboard input corresponding to the ghost user interface element.

16. A method for replacing a first user interface element created by a first application by a first substitute user interface element created by a scheduled code path in a multithreaded computing environment, the method comprising the steps of:
detecting a first flip-window signal while the first user interface element is being displayed;
creating, responsively to the first flip-window signal, the first substitute user interface element in the context of the scheduled code path;
superimposing the first substitute user interface element over a first-user-interface-element-occupied real estate; and
caching input corresponding to the first substitute user interface element wherein the cached input is subsequently handled by the first application.

17. The method of claim 16 wherein the step of creating includes creating the scheduled code path if the scheduled code path does not exist.

18. The method of claim 16, the method further having the step of replacing a second user interface element created by a second application by a second substitute user interface element created by the scheduled code path responsively to a second flip-window signal.

19. The method of claim 16, the method further having the step of painting over the first user interface element responsively to the first flip-window signal.

A3

20. The method of claim 16 wherein the step of forwarding includes input comprising a plurality of events and/or a plurality of messages.

21. The method of claim 17, the method further having the step of caching input corresponding to the application for forwarding to the application in response to a first flop-window signal.

22. The method of claim 18, the method further having the steps of selecting and discarding at least some of the cached input.

23. The method of claim 19 wherein the step of forwarding includes forwarding the input to the scheduled code path.

24. The method of claim 20, the method further having the step of handling at least some of the input by the scheduled code path.

25. The method of claim 24, the method further having the step of the scheduled code path handling at least one event in the input by terminating the first application.

26. The method of claim 16, the method further having the step of placing a special message/event in a queue for the first application to generate a flop-window signal.

27. The method of claim 16 wherein the scheduled code path is a thread.

28. A method of using a designated scheduled code path for providing substitute user interfaces for replacing user interfaces corresponding to a plurality of scheduled code paths, the conprising the steps of:

generating a first flip-window signal corresponding to a first scheduled code path;

generating a second flip-window signal corresponding to a second secheduled code path;

replacing, responsively to the first flip-window signal, a first window controlled by the first scheduled path with a first substitute window controlled by the designated scheduled code path; and

replacing, responsively to the second flip-window signal, a second window controlled by the second scheduled path with a second substitute window controlled by the designated scheduled code path.

29. The method of claim 28, the method further having the steps of generating a first flop-window signal; and replacing, responsively to the first flop-window signal, a first substitute window controlled by the designated scheduled code path with the first window controlled by the first scheduled path.

30. A multithreaded computing system for executing a first application having a first user interface, wherein if the first application is non-responsive to user input, the first user interface is replaced by a first ghost user interface, the system comprising:

a non-responsive-application detecting code for detecting a non-responsive application;

a ghost thread for creating the first ghost user interface to replace the first user interface responsively to detection of a non-responsive application;

a high priority special entry in a queue for the first application for detecting if the first application is responsive; and

a plurality of computer executable instructions for destroying the first ghost user interface responsively to handling of the high priority special entry by the first application.

31. The system of claim 30 further having a cache of events and messages for handling by the first application wherein the events and messages are directed at the ghost thread.

32. The system of claim 30 further having a second non-responsive application with a second user interface and a second ghost user interface created by the ghost thread.

A5

33. The system of claim 30 further having a responsive-application detecting code for detecting when a non-responsive application becomes responsive.

34. The system of claim 33 further having a responsive-application user interface restoring code for replacing the first ghost user interface with the first user interface responsively to detecting that the first application has become responsive.

35. A computer readable media having computer executable instructions for carrying out the steps of a method for managing a user interface in a multithreaded computing environment, the user interface comprising a plurality of user interface elements wherein a first user interface element in the plurality of user interface elements corresponds to a first application, wherein furthermore the first application having a first thread having control over the first user interface element, method comprising the steps of:
signaling a hung state for the first application; (if there is a delay greater than a predetermined threshold in handling events queued in an event queue for handling by the first application);
creating a ghost user interface element, with a ghost thread, responsively to a hung signal indicating the first application is in the hung state wherein the ghost user interface element replaces the first user interface element in the user interface;
placing a high priority special event in the event queue for handling by the first application wherein handling of the special event generates a activity-detect message; and

detecting the wakeup message and responsively to the activity-detect message replacing the ghost user interface element by the first user interface element.